Raymond Salzmann

EEL3744C

11/28/17

Pre-Lab Questions

1. What is the correlation between the amount of data points used to recreate the waveform and the overall quality of the waveform?

   The more data points that are used to recreate the waveform, the better the overall quality of the waveform will be, as there will be less distance between the individual data points.

2. How many DMA channels are available on the XMEGA?

   There are 4 DMA channels available on the XMEGA.

3. How many different base value trigger options are available within the XMEGA's DMA system?

   There are 27 different base value trigger option available within the XMEGA's DMA system.

Problems Encountered

I had some issues with the order in which I initialized the different register for the DMA. Also had some issues with the speaker.

Future Work/Applications

If I had more time to work on the lab I would create some header files and such. This lab is particularly useful for recreating waveforms and has applications in signals & systems.

Pre-Lab

Part A:

Pseudocode:

1. Initialize the clock to 32MHz
2. Create the lookup table of values with 0x0FFF as the maximum value
3. Use a counter to iterate through the lookup table with an ISR going off the TCC0
4. Output the data using the DACA_CH0 on Port A Pin 2

Program Code:

```
/* Lab 7 Part A
* Name: Raymond Salzmann
*Section #: 2B04
* Description: Generating Waveform with lookup table
* TA Name: Keith Fitzgerald
*/

#include <avr/io.h>
#include <avr/interrupt.h>
void CLK_INIT(void);
void DAC_INIT(void);
void TC_INIT(void);
//lookup table acquired from the Internet for 256 values
int table[] = {0x800,0x832,0x864,0x896,0x8c8,0x8fa,0x92c,0x95e,
       0x98f,0x9c0,0x9f1,0xa22,0xa52,0xa82,0xab1,0xae0,
       0xb0f,0xb3d,0xb6b,0xb98,0xbc5,0xbf1,0xc1c,0xc47,
       0xc71,0xc9a,0xcc3,0xceb,0xd12,0xd39,0xd5f,0xd83,
       0xda7,0xdca,0xded,0xe0e,0xe2e,0xe4e,0xe6c,0xe8a,
       0xea6,0xec1,0xedc,0xef5,0xf0d,0xf24,0xf3a,0xf4f,
       0xf63,0xf76,0xf87,0xf98,0xfa7,0xfb5,0xfc2,0xfcd,
       0xfd8,0xfe1,0xfe9,0xff0,0xff5,0xff9,0xffd,0xffe,
       0xfff,0xffe,0xffd,0xff9,0xff5,0xff0,0xfe9,0xfe1,
       0xfd8,0xfcd,0xfc2,0xfb5,0xfa7,0xf98,0xf87,0xf76,
       0xf63,0xf4f,0xf3a,0xf24,0xf0d,0xef5,0xedc,0xec1,
```

```
	0xea6,0xe8a,0xe6c,0xe4e,0xe2e,0xe0e,0xded,0xdca,
	0xda7,0xd83,0xd5f,0xd39,0xd12,0xceb,0xcc3,0xc9a,
	0xc71,0xc47,0xc1c,0xbf1,0xbc5,0xb98,0xb6b,0xb3d,
	0xb0f,0xae0,0xab1,0xa82,0xa52,0xa22,0x9f1,0x9c0,
	0x98f,0x95e,0x92c,0x8fa,0x8c8,0x896,0x864,0x832,
	0x800,0x7cd,0x79b,0x769,0x737,0x705,0x6d3,0x6a1,
	0x670,0x63f,0x60e,0x5dd,0x5ad,0x57d,0x54e,0x51f,
	0x4f0,0x4c2,0x494,0x467,0x43a,0x40e,0x3e3,0x3b8,
	0x38e,0x365,0x33c,0x314,0x2ed,0x2c6,0x2a0,0x27c,
	0x258,0x235,0x212,0x1f1,0x1d1,0x1b1,0x193,0x175,
	0x159,0x13e,0x123,0x10a,0xf2,0xdb,0xc5,0xb0,
	0x9c,0x89,0x78,0x67,0x58,0x4a,0x3d,0x32,
	0x27,0x1e,0x16,0xf,0xa,0x6,0x2,0x1,
	0x0,0x1,0x2,0x6,0xa,0xf,0x16,0x1e,
	0x27,0x32,0x3d,0x4a,0x58,0x67,0x78,0x89,
	0x9c,0xb0,0xc5,0xdb,0xf2,0x10a,0x123,0x13e,
	0x159,0x175,0x193,0x1b1,0x1d1,0x1f1,0x212,0x235,
	0x258,0x27c,0x2a0,0x2c6,0x2ed,0x314,0x33c,0x365,
	0x38e,0x3b8,0x3e3,0x40e,0x43a,0x467,0x494,0x4c2,
	0x4f0,0x51f,0x54e,0x57d,0x5ad,0x5dd,0x60e,0x63f,
	0x670,0x6a1,0x6d3,0x705,0x737,0x769,0x79b,0x7cd};

	int count = 0;

ISR(TCC0_OVF_vect){
	if(count > 255){
		count = 0; //start the count over
	}
	while(DACA_STATUS != 0x03);

	TCC0_INTFLAGS = 0x01; //clear the overflow flag

	DACA_CH0DATA = table[count]; //set new output for DAC

	count++;

	return;
}


int main(void)
{
	CLK_INIT();
	DAC_INIT();
	TC_INIT();

	while(1);

	return 0;
}

void CLK_INIT(void)
{
	OSC_CTRL = 0x02;

	while(!(OSC_STATUS & 0x02));

	CPU_CCP = 0xD8;
```

```
	CLK_CTRL = 0x01;
}

void DAC_INIT(void){
	DACA_CTRLC = 0x18; //Setting the reference to Port B AREF
	DACA_CTRLA = 0x05; //Setting the DAC to output to channel 0
}

void TC_INIT(void){
		TCC0_CTRLA = 0x01; //Timer setting clk

		TCC0_INTCTRLA= 0x01; //low overflow interrupt

		TCC0_PER = 0x01AA; //Approximately the time for 300 Hz

		PMIC_CTRL = 0x01; //Enable low level interrupts

		sei();
}
```

Part B:

Pseudocode:

1. Keep code from Part A
2. Initialize the DMA system with special care to the value of the TRFCNT register
3. Make sure for the initialization of the DMA system that the SRCADDR and DESTADDR register are initialized in the correct order

Program Code:

```
/* Lab 7 Part B
* Name: Raymond Salzmann
*Section #: 2B04
* Description: Generating Waveform with DMA
* TA Name: Keith Fitzgerald
*/

#include <avr/io.h>
#include <avr/interrupt.h>
void CLK_INIT(void);
void DAC_INIT(void);
void TC_INIT(void);
void DMA_INIT(void);

ISR(TCC0_OVF_vect){
```

```
	TCC0_INTFLAGS = 0x01; //Clear overflow flag
	return;
}

//lookup table acquired from the Internet for 256 values
int table[] = {0x800,0x832,0x864,0x896,0x8c8,0x8fa,0x92c,0x95e,
	0x98f,0x9c0,0x9f1,0xa22,0xa52,0xa82,0xab1,0xae0,
	0xb0f,0xb3d,0xb6b,0xb98,0xbc5,0xbf1,0xc1c,0xc47,
	0xc71,0xc9a,0xcc3,0xceb,0xd12,0xd39,0xd5f,0xd83,
	0xda7,0xdca,0xded,0xe0e,0xe2e,0xe4e,0xe6c,0xe8a,
	0xea6,0xec1,0xedc,0xef5,0xf0d,0xf24,0xf3a,0xf4f,
	0xf63,0xf76,0xf87,0xf98,0xfa7,0xfb5,0xfc2,0xfcd,
	0xfd8,0xfe1,0xfe9,0xff0,0xff5,0xff9,0xffd,0xffe,
	0xfff,0xffe,0xffd,0xff9,0xff5,0xff0,0xfe9,0xfe1,
	0xfd8,0xfcd,0xfc2,0xfb5,0xfa7,0xf98,0xf87,0xf76,
	0xf63,0xf4f,0xf3a,0xf24,0xf0d,0xef5,0xedc,0xec1,
	0xea6,0xe8a,0xe6c,0xe4e,0xe2e,0xe0e,0xded,0xdca,
	0xda7,0xd83,0xd5f,0xd39,0xd12,0xceb,0xcc3,0xc9a,
	0xc71,0xc47,0xc1c,0xbf1,0xbc5,0xb98,0xb6b,0xb3d,
	0xb0f,0xae0,0xab1,0xa82,0xa52,0xa22,0x9f1,0x9c0,
	0x98f,0x95e,0x92c,0x8fa,0x8c8,0x896,0x864,0x832,
	0x800,0x7cd,0x79b,0x769,0x737,0x705,0x6d3,0x6a1,
	0x670,0x63f,0x60e,0x5dd,0x5ad,0x57d,0x54e,0x51f,
	0x4f0,0x4c2,0x494,0x467,0x43a,0x40e,0x3e3,0x3b8,
	0x38e,0x365,0x33c,0x314,0x2ed,0x2c6,0x2a0,0x27c,
	0x258,0x235,0x212,0x1f1,0x1d1,0x1b1,0x193,0x175,
	0x159,0x13e,0x123,0x10a,0xf2,0xdb,0xc5,0xb0,
	0x9c,0x89,0x78,0x67,0x58,0x4a,0x3d,0x32,
	0x27,0x1e,0x16,0xf,0xa,0x6,0x2,0x1,
	0x0,0x1,0x2,0x6,0xa,0xf,0x16,0x1e,
	0x27,0x32,0x3d,0x4a,0x58,0x67,0x78,0x89,
	0x9c,0xb0,0xc5,0xdb,0xf2,0x10a,0x123,0x13e,
	0x159,0x175,0x193,0x1b1,0x1d1,0x1f1,0x212,0x235,
	0x258,0x27c,0x2a0,0x2c6,0x2ed,0x314,0x33c,0x365,
	0x38e,0x3b8,0x3e3,0x40e,0x43a,0x467,0x494,0x4c2,
	0x4f0,0x51f,0x54e,0x57d,0x5ad,0x5dd,0x60e,0x63f,
	0x670,0x6a1,0x6d3,0x705,0x737,0x769,0x79b,0x7cd};

int main(void)
{
	CLK_INIT();
	DAC_INIT();
	TC_INIT();
	DMA_INIT();

	while(1);

	return 0;
}

void CLK_INIT(void)
{
	OSC_CTRL = 0x02;

	while(!(OSC_STATUS & 0x02));

	CPU_CCP = 0xD8;
```

```
	CLK_CTRL = 0x01;
}

void DAC_INIT(void){
	DACA_CTRLC = 0x18; //Setting the reference to Port B AREF
	DACA_CTRLA = 0x05; //Setting the DAC to output to channel 0
}

void TC_INIT(void){
	TCC0_CTRLA = 0x01; //Timer setting clk

	TCC0_PER = 0x01AA; //Approximately the time for 300 Hz

	TCC0_INTCTRLA= 0x01; //low overflow interrupt

	PMIC_CTRL = 0x01; //Enable low level interrupts

	sei();
}

void DMA_INIT(void){
		int16_t addr =(int16_t) & table;
		int16_t dest =(int16_t) & DACA_CH0DATA;

		DMA_CTRL = DMA_ENABLE_bm |
			DMA_PRIMODE_CH0123_gc; //0x83

		DMA_CH0_REPCNT= 0x00;
		DMA_CH0_CTRLA= 0xA5;//Enabled,Repeat,Single,Burst 2 bytes

		DMA_CH0_ADDRCTRL= DMA_CH_SRCRELOAD_BLOCK_gc |
			DMA_CH_SRCDIR_INC_gc |
			DMA_CH_DESTRELOAD_BURST_gc |
			DMA_CH_DESTDIR_INC_gc ; //0x59

		DMA_CH0_TRIGSRC= 0x40;
		DMA_CH0_TRFCNT= 0x01FE;

		DMA_CH0_SRCADDR0 = addr;

		addr = addr >> 8;
		DMA_CH0_SRCADDR1 = addr;

		addr = addr >> 8;
		DMA_CH0_SRCADDR2 = addr;

		DMA_CH0_DESTADDR0 = dest;

		dest = dest >> 8;
		DMA_CH0_DESTADDR1 = dest;

		dest = dest >> 8;
		DMA_CH0_DESTADDR2= dest;

		return;
}
```

Part D:

Pseudocode:

1. Copy Code from Part B
2. Set up USART
3. Initialize speaker to turn on
4. Profit

Program Code:

```
/* Lab 7 Part D
* Name: Raymond Salzmann
*Section #: 2B04
* Description: Creating Synthesizer Keyboard
* TA Name: Keith Fitzgerald
*/

#include <avr/io.h>
#include <avr/interrupt.h>

void CLK_INIT(void);
void DAC_INIT(void);
void TC_INIT(int period);
void DMA_INIT(int waveform);
char IN_CHAR(void);
void OUT_CHAR(char character);
void OUT_STRING(char* string);
void SPEAKER_INIT(void);
void USART_INIT(void);

ISR(TCC0_OVF_vect){
	TCC0_INTFLAGS = 0x01; //Clear overflow flag
	return;
}

//lookup table acquired from the Internet for 256 values
int sinusoid[] = {0x800,0x832,0x864,0x896,0x8c8,0x8fa,0x92c,0x95e,
	0x98f,0x9c0,0x9f1,0xa22,0xa52,0xa82,0xab1,0xae0,
	0xb0f,0xb3d,0xb6b,0xb98,0xbc5,0xbf1,0xc1c,0xc47,
	0xc71,0xc9a,0xcc3,0xceb,0xd12,0xd39,0xd5f,0xd83,
	0xda7,0xdca,0xded,0xe0e,0xe2e,0xe4e,0xe6c,0xe8a,
	0xea6,0xec1,0xedc,0xef5,0xf0d,0xf24,0xf3a,0xf4f,
	0xf63,0xf76,0xf87,0xf98,0xfa7,0xfb5,0xfc2,0xfcd,
	0xfd8,0xfe1,0xfe9,0xff0,0xff5,0xff9,0xffd,0xffe,
	0xfff,0xffe,0xffd,0xff9,0xff5,0xff0,0xfe9,0xfe1,
	0xfd8,0xfcd,0xfc2,0xfb5,0xfa7,0xf98,0xf87,0xf76,
```

```
        0xf63,0xf4f,0xf3a,0xf24,0xf0d,0xef5,0xedc,0xec1,
        0xea6,0xe8a,0xe6c,0xe4e,0xe2e,0xe0e,0xded,0xdca,
        0xda7,0xd83,0xd5f,0xd39,0xd12,0xceb,0xcc3,0xc9a,
        0xc71,0xc47,0xc1c,0xbf1,0xbc5,0xb98,0xb6b,0xb3d,
        0xb0f,0xae0,0xab1,0xa82,0xa52,0xa22,0x9f1,0x9c0,
        0x98f,0x95e,0x92c,0x8fa,0x8c8,0x896,0x864,0x832,
        0x800,0x7cd,0x79b,0x769,0x737,0x705,0x6d3,0x6a1,
        0x670,0x63f,0x60e,0x5dd,0x5ad,0x57d,0x54e,0x51f,
        0x4f0,0x4c2,0x494,0x467,0x43a,0x40e,0x3e3,0x3b8,
        0x38e,0x365,0x33c,0x314,0x2ed,0x2c6,0x2a0,0x27c,
        0x258,0x235,0x212,0x1f1,0x1d1,0x1b1,0x193,0x175,
        0x159,0x13e,0x123,0x10a,0xf2,0xdb,0xc5,0xb0,
        0x9c,0x89,0x78,0x67,0x58,0x4a,0x3d,0x32,
        0x27,0x1e,0x16,0xf,0xa,0x6,0x2,0x1,
        0x0,0x1,0x2,0x6,0xa,0xf,0x16,0x1e,
        0x27,0x32,0x3d,0x4a,0x58,0x67,0x78,0x89,
        0x9c,0xb0,0xc5,0xdb,0xf2,0x10a,0x123,0x13e,
        0x159,0x175,0x193,0x1b1,0x1d1,0x1f1,0x212,0x235,
        0x258,0x27c,0x2a0,0x2c6,0x2ed,0x314,0x33c,0x365,
        0x38e,0x3b8,0x3e3,0x40e,0x43a,0x467,0x494,0x4c2,
        0x4f0,0x51f,0x54e,0x57d,0x5ad,0x5dd,0x60e,0x63f,
        0x670,0x6a1,0x6d3,0x705,0x737,0x769,0x79b,0x7cd};

int saw[] = {0x00,0x10,0x20,0x30,0x40,0x50,0x60,0x70,0x80, 0x90,0xa0,0xb0,
        0xc0,0xd0,0xe0,0xf0,0x100,0x110,0x120,0x130,0x140,0x150,0x160,0x170,0x180,
        0x190,0x1a0,0x1b0,0x1c0,0x1d0,0x1e0,0x1f0,0x200,0x210,0x220,0x230,0x240,0x250,
        0x260,0x270,0x280,0x290,0x2a0,0x2b0,0x2c0,0x2d0,0x2e0,0x2f0,0x300,
        0x310,0x320,0x330,0x340,0x350,0x360,0x370,0x380,0x390,0x3a0,0x3b0,0x3c0,0x3d0,
        0x3e0,0x3f0,0x400,0x410,0x420,0x430,0x440,0x450,0x460,0x470,0x480,
        0x490,0x4a0,0x4b0,0x4c0,0x4d0,0x4e0,0x4f0,0x500,0x510,0x520,0x530,0x540,0x550,
        0x560,0x570,0x580,0x590,0x5a0,0x5b0,0x5c0,0x5d0,0x5e0,0x5f0,0x600,
        0x610,0x620,0x630,0x640,0x650,0x660,0x670,0x680,0x690,0x6a0,0x6b0,0x6c0,0x6d0,
        0x6e0,0x6f0,0x700,0x710,0x720,0x730,0x740,0x750,0x760,0x770,0x780,
        0x790,0x7a0,0x7b0,0x7c0,0x7d0,0x7e0,0x7f0,0x800,0x80f,0x81f,0x82f,0x83f,0x84f,
        0x85f,0x86f,0x87f,0x88f,0x89f,0x8af,0x8bf,0x8cf,0x8df,0x8ef,0x8ff,
        0x90f,0x91f,0x92f,0x93f,0x94f,0x95f,0x96f,0x97f,0x98f,0x99f,0x9af,0x9bf,0x9cf,
        0x9df,0x9ef,0x9ff,0xa0f,0xa1f,0xa2f,0xa3f,0xa4f,0xa5f,0xa6f,0xa7f,
        0xa8f,0xa9f,0xaaf,0xabf,0xacf,0xadf,0xaef,0xaff,0xb0f,0xb1f,0xb2f,0xb3f,0xb4f,
        0xb5f,0xb6f,0xb7f,0xb8f,0xb9f,0xbaf,0xbbf,0xbcf,0xbdf,0xbef,0xbff,0xc0f,0xc1f,
0xc2f, 0xc3f,0xc4f,0xc5f,0xc6f,0xc7f,0xc8f,0xc9f,0xcaf,0xcbf,0xccf,0xcdf,
        0xcef,0xcff,0xd0f,0xd1f,0xd2f,0xd3f,0xd4f,0xd5f,0xd6f,0xd7f,
        0xd8f,0xd9f,0xdaf,0xdbf,0xdcf,0xddf,0xdef,0xdff,0xe0f,0xe1f,0xe2f,0xe3f,
        0xe4f,0xe5f,0xe6f,0xe7f,0xe8f,0xe9f,0xeaf,0xebf,0xecf,0xedf,0xeef,0xeff,
        0xf0f,0xf1f,0xf2f,0xf3f,0xf4f,0xf5f,0xf6f,0xf7f,0xf8f,0xf9f,0xfaf,0xfbf,0xfcf,
        0xfdf,0xfef,0xfff};

int choose = 1;

int main(void)
{
        char choice;
        CLK_INIT();
        USART_INIT();
        DAC_INIT();
        SPEAKER_INIT();
        DMA_INIT(1);

        choose = 0;
```

```
while(1){
	choice = IN_CHAR();
	OUT_CHAR(choice);
	OUT_CHAR(' ');
	switch(choice){
		case 's':
		case 'S':
		DMA_CH0_CTRLA = DMA_CH_RESET_bm;//resets the DMA

		if (choose == 1){
			DMA_INIT(1);
			choose = 0;
		}

		else{
			DMA_INIT(0);
			choose = 1;
		}

		break;

		case 'w':
		case 'W':
		TC_INIT(0x007A); //1046.50 Hz
		for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
		break;

		case '3':
		TC_INIT(0x0073); //1108.73 Hz
		for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
		break;

		case 'e':
		case 'E':
		TC_INIT(0x006D); //1174.66 Hz
		for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
		break;

		case '4':
		TC_INIT(0x0066); //1244.51 Hz
		for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
		break;

		case 'r':
		case 'R':
		TC_INIT(0x0061); //1318.51 Hz
		for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
		break;

		case 't':
		case 'T':
		TC_INIT(0x005B); //1396.91 Hz
		for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
		break;

		case '6':
		TC_INIT(0x0056); //1479.98 Hz
```

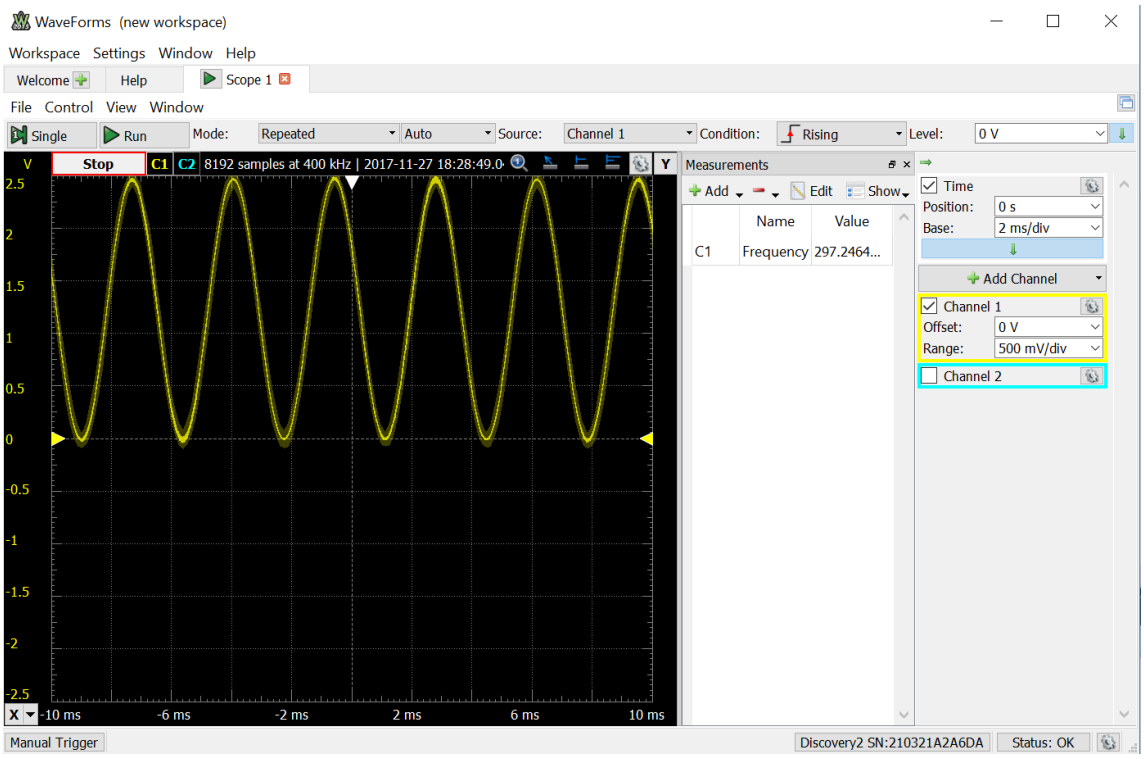
```
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case 'y':
                        case 'Y':
                        TC_INIT(0x0051); //1567.98 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case '7':
                        PORTC_OUTSET = 0x80;
                        TC_INIT(0x004C); //1661.22 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case 'u':
                        case 'U':
                        TC_INIT(0x0048); //1760.00 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case '8':
                        TC_INIT(0x0044); //1864.66 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case 'i':
                        case 'I':
                        TC_INIT(0x0040); //1975.53 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case 'o':
                        case 'O':
                        TC_INIT(0x003C); //2093.00 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;

                        case '9':
                        TC_INIT(0x0039); //2217.46 Hz
                        for (int32_t ii = 0; ii < 0x0001FFFF; ii++);
                        break;
                }
                TCC0_CTRLA = 0x00;
        }
        return 0;
}

void CLK_INIT(void)
{
        OSC_CTRL = 0x02;

        while(!(OSC_STATUS & 0x02));

        CPU_CCP = 0xD8;

        CLK_CTRL = 0x01;
}
```

```
void DAC_INIT(void){
	DACA_CTRLC = 0x18; //Setting the reference to Port B AREF
	DACA_CTRLB = 0x20;
	DACA_CTRLA = 0x09; //Setting the DAC to output to channel 1
}

void TC_INIT(int16_t period){
	TCC0_CTRLA = 0x01; //Timer setting clk

	//TCC0_PER = 0x01AA; //Approximately the time for 300 Hz

	TCC0_PER = period; //generates desired frequency based on key pressed

	TCC0_INTCTRLA= 0x01; //low overflow interrupt

	PMIC_CTRL = 0x01; //Enable low level interrupts

	sei();
}

void DMA_INIT(int waveform){
	int16_t addr;
	int16_t dest =(int16_t) & DACA_CH1DATA;

	if (waveform == 1){
		addr = (int16_t) & sinusoid;
	}

	if (waveform == 0){
		addr = (int16_t) & saw;
	}

	DMA_CTRL = DMA_ENABLE_bm |
	DMA_PRIMODE_CH0123_gc; //0x83

	DMA_CH0_REPCNT= 0x00;
	DMA_CH0_CTRLA= 0xA5;//Enabled,Repeat,Single,Burst 2 bytes

	DMA_CH0_ADDRCTRL= DMA_CH_SRCRELOAD_BLOCK_gc |
	DMA_CH_SRCDIR_INC_gc |
	DMA_CH_DESTRELOAD_BURST_gc |
	DMA_CH_DESTDIR_INC_gc ; //0x59

	DMA_CH0_TRIGSRC= 0x40;
	DMA_CH0_TRFCNT= 0x01FE;

	DMA_CH0_SRCADDR0 = addr;

	addr = addr >> 8;
	DMA_CH0_SRCADDR1 = addr;

	addr = addr >> 8;
	DMA_CH0_SRCADDR2 = addr;

	DMA_CH0_DESTADDR0 = dest;

	dest = dest >> 8;
```

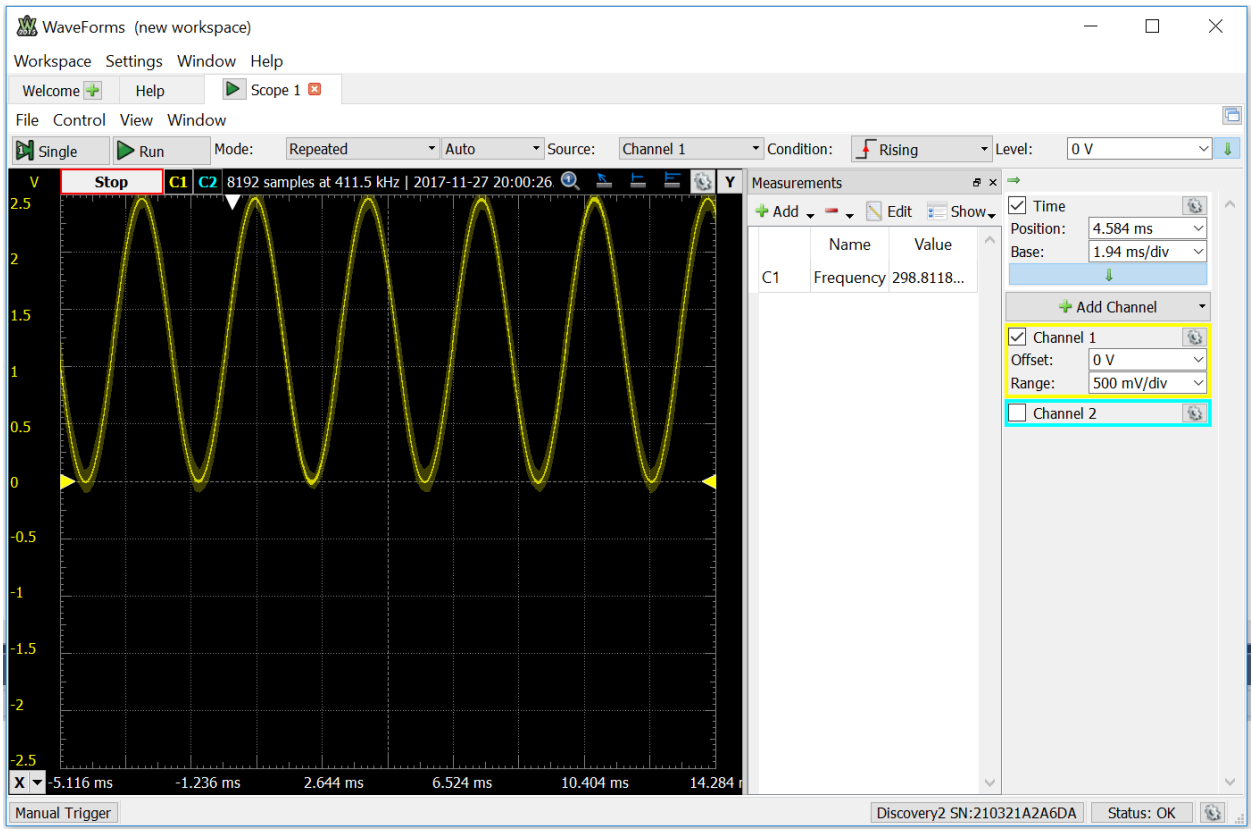
```
	DMA_CH0_DESTADDR1 = dest;

	dest = dest >> 8;
	DMA_CH0_DESTADDR2= dest;

	return;
}

void USART_INIT(void)
{
	PORTD_DIRCLR = 0x04;
	PORTD_DIRSET = 0x08;
	USARTD0_BAUDCTRLA = 0x01; //sets baud rate to 1 MHz
	USARTD0_BAUDCTRLB = 0x00;
	USARTD0.CTRLC = USART_CMODE_ASYNCHRONOUS_gc | USART_PMODE_DISABLED_gc |
	USART_SBMODE_bp | USART_CHSIZE_8BIT_gc;
	USARTD0.CTRLB = USART_RXEN_bm | USART_TXEN_bm;
	return;
}

void OUT_CHAR(char character)
{

	while (!(USARTD0_STATUS & USART_DREIF_bm));

	USARTD0_DATA = character;

	return;
}

char IN_CHAR(void)
{
	while (!(USARTD0_STATUS & USART_RXCIF_bm));

	return USARTD0_DATA;
}

void OUT_STRING(char* string)
{
	while(*string)
	{
		OUT_CHAR(*string);
		string++;
	}
	return;
}


void SPEAKER_INIT(void){
	PORTA_DIRSET = 0x08;

//	PORTA_DIRSET = 0x24;
//	PORTA_OUT = 0x20;

	PORTC_DIRSET = 0x80;
	PORTC_OUTSET = 0x80;

	return;
```

}

Appendix



Screenshot for Part A



Screenshot for Part B